MARLeME: A Multi-Agent Reinforcement Learning Model Extraction Library

1st Dmitry Kazhdan
Computer Science and Technology
The University of Cambridge
Cambridge, UK
dk525@cam.ac.uk

2nd Zohreh Shams
Computer Science and Technology
The University of Cambridge
Cambridge, UK
zohreh.shams@cl.cam.ac.uk

3rd Pietro Lio
Computer Science and Technology
The University of Cambridge
Cambridge, UK
pietro.lio@cl.cam.ac.uk

***Abstract*—Multi-Agent Reinforcement Learning (MARL) encompasses a powerful class of methodologies that have been applied in a wide range of fields. An effective way to further empower these methodologies is to develop approaches and tools that could expand their interpretability and explainability. In this work, we introduce MARLeME: a MARL model extraction library, designed to improve explainability of MARL systems by approximating them with symbolic models. Symbolic models offer a high degree of interpretability, well-defined properties, and verifiable behaviour. Consequently, they can be used to inspect and better understand the underlying MARL systems and corresponding MARL agents, as well as to replace all/some of the agents that are particularly safety and security critical. In this work, we demonstrate how MARLeME can be applied to two well-known case studies (Cooperative Navigation and RoboCup Takeaway), using extracted models based on Abstract Argumentation.**

***Index Terms*—Multi-Agent Reinforcement Learning, Interpretability, Abstract Argumentation, Explainability, Model Extraction, Knowledge Extraction, Symbolic Reasoning, Library**

I. Introduction

Multi-Agent Reinforcement Learning (MARL) has achieved groundbreaking results in a wide range of fields, and is currently a highly active research area of Machine Learning (ML) [6], [38], [39]. MARL deals with teams of agents that learn how to act optimally in stochastic environments through trial-and-error, and has been successfully applied to tasks requiring cooperative/competitive multi-agent behaviour, such as large-scale fleet management [31], swarm systems [32], and task allocation [30].

Unfortunately, a substantial amount of recent MARL approaches represent decision-making policies using very complex models, such as Deep Neural Networks (DNNs) [28], making it extremely challenging to directly understand an agent's action-selection strategy. A lack of interpretability of such systems leads to a lack of confidence in the correctness of their behaviour, which is crucial in safety-critical applications, such as self-driving cars or healthcare. Furthermore, this lack of interpretability implies that optimal strategies learned by the MARL agents cannot be used to improve our understanding of the corresponding domain. Approaches based on symbolic reasoning, on the other hand, offer interpretable, verifiable models with well-defined properties. As a result, there has recently been increasing interest in approaches capable of combining symbolic reasoning with ML [26], [27], [29].

One technique that allows reaping the benefits of both ML-based and symbolic approaches is *model extraction* [36]. Model extraction refers to approaches that approximate a complex model (e.g. a DNN) with a simpler, interpretable one (e.g. a rule-based model), facilitating understanding of the complex model. Intuitively, statistical properties of the complex model should be reflected in the extracted model, provided approximation quality of the extracted model (referred to as *fidelity*) is high enough.

This paper introduces MARLeME: a (M)ulti-(A)gent (R)einforcement (Le)arning (M)odel (E)xtraction library, designed to improve interpretability of MARL systems using model extraction. MARLeME is an open-source [1], easy-to-use, plug-and-play library, that can be seamlessly integrated with a wide range of existing MARL tasks. To the best of our knowledge, this is the first open-source MARL library focusing on interpretable model extraction from MARL systems.

The extracted models introduced in this paper are based on Abstract Argumentation (AA). To the best of our knowledge, this is the first time AA-based models have been used for MARL agent approximation, despite the advantages of AA-based systems (discussed further in Section III-B).

The rest of this paper is structured as follows: Section II reviews work related to RL model extraction. Section III discusses the relevant background. Section IV gives an overview of the MARLeME library and the AA models. Section V and Section VI present an evaluation of MARLeME using two RL benchmarks (Cooperative Navigation and RoboCup Takeaway). Finally, Section VII gives some concluding remarks.

II. Related Work

XAI: Explainable AI (XAI) is a field focusing on the explainability/trustworthiness of ML systems, and has recently seen a surge of interest, with a wide range of approaches proposed in recent years (a comprehensive survey can be found in [40]). As discussed in Section I, our work focuses on model extraction, and can thus be categorised as a model-agnostic,

[1] https://github.com/dmitrykazhdan/MARLeME

global interpretability-based approach (a discussion of alternative approaches to XAI and their relative merits/drawbacks can be found in [40]). Furthermore, our work focuses on explainability of multi-agent systems (a comprehensive survey of XAI approaches in the context of multi-agent systems can be found in [41]). Crucially, explainability of MARL systems is seldom addressed in existing work on explainability, with focus being predominantly on supervised learning approaches instead.

SARL Model Extraction: A range of recent work has focused on Single-Agent Reinforcement Learning (SARL) model extraction [21], [24], [25], relying on different types of extracted models. Unlike our work, which focuses on teams of RL agents, all of the above approaches were designed for and evaluated using the single RL agent case only. However, extensions of these approaches to the multi-agent case could in principle be incorporated into MARLeME as new extracted model types.

Imitation Learning: Our work is also partially related to imitation learning [22], [23]. Similarly to MARLeME, imitation learning approaches derive action-selection policies from input trajectories. However, imitation learning approaches focus on the quality of the policy (i.e. its performance on the corresponding task), without paying attention to its interpretability. Achieving the latter while preserving performance is a unique focus of MARLeME.

III. Background

A. Reinforcement Learning

RL is applied to tasks that require automated decision making, such as autonomous navigation [7], healthcare [8], and many others [9]. This work only requires a high-level understanding of key RL concepts, that are described below. Further details can be found in [35].

1) Single-Agent Reinforcement Learning: SARL focuses on a single autonomous agent that interacts with a given environment over time, learning how to act optimally in this (possibly stochastic) environment through trial-and-error. At each time step t (over a sequence of timesteps), the agent receives a representation of its environment in the form of a state s_t, and selects an action a_t on the basis of that state. In the next time step, the agent receives a numerical reward from the environment r_t, and transitions to the next state s_{t+1}, according to the environment dynamics. This is repeated for a sequence of time steps, until a terminal state is reached. The goal of a RL agent is to find an *optimal policy* (a mapping from input states to actions) that maximises its long-term cumulative reward. A sequence of states and corresponding selected actions $[(s_0, a_0), ..., (s_T, a_T)]$ of an agent from a start state to a terminal state is referred to as an *episode sequence*. A set of such sequences collected from multiple episodes is referred to as a *trajectory*. Further details regarding RL theory may be found here [28], [35].

2) Multi-Agent Reinforcement Learning: MARL incorporates RL into Multi-Agent Systems (MASs), and consists of teams of RL agents that must compete or cooperate in order to solve complex tasks. MARL has been successfully applied in a range of tasks that rely on automated decision making in multi-agent systems, such as controlling power flow in an electrical power-grid [3], making economic decisions [4], and many others [5]. Further details regarding the challenges and opportunities of MARL, as well as the mathematical details, may be found in [6].

B. Abstract Argumentation

Extracted models presented in this paper rely on Abstract Argumentation (AA) [19], due to the popularity of AA in autonomous agent and multi-agent decision making [16]–[18]. Argumentation allows reasoning in presence of incomplete and inconsistent knowledge, which is often the case in multi-agent settings, where the agents are unlikely to have complete knowledge of the environment and may have conflicting objectives. In addition, AA-based decision-making models are interpretable and thus highly suitable for model extraction. Despite these advantages, to the best of our knowledge, this is the first time that AA-based models have been used for interpretable model extraction in RL.

In this work we make use of Value-based Argumentation Frameworks (VAFs) which are based on Abstract Argumentation Frameworks (AFs). Informally, an $AF = (Arg, Att)$, consists of a set of arguments Arg, which are defeasible rules of inference, and a binary attack relation between arguments $Att \subseteq Arg \times Arg$ that represent conflicts between arguments. In a value-based argumentation framework $VAF = (Arg, Att, V, val, Valpref)$, there are values attached to arguments that represent their relative utility and hence dictate which argument is preferred to another in face of conflict: V is a set of possible argument values , $val : Arg \rightarrow V$ is a function that assigns values to arguments, and $Valpref$ is an ordering over these values. Naturally, argumentation frameworks can be represented as directed graphs, where nodes are arguments and an edge from node A to B represents an attack from the former to the latter.

When representing knowledge using a VAF, it is possible to determine which arguments in the VAF are 'winning' or optimal, by defining suitable *semantics* (sets of well-defined rules) that specify whether an argument should be *accepted* (i.e. treated as optimal). The set of accepted arguments of a VAF is referred to as its *extension*. In this work, we make use of the *grounded extension* (GE) semantics when computing acceptable arguments from a VAF, that will be described in the remainder of this section. Details regarding other extension types may be found in [19].

Given an AF (Arg, Att), suppose $S \subseteq Arg$, then:

- S attacks an argument A iff some member of S attacks A
- S is *conflict-free* iff S attacks none of its members
- S *defends* an argument A iff S attacks all arguments attacking A
- S is an *admissible extension* of an AF iff S is conflict-free and defends all its elements

- S is a *complete extension* of an AF iff it is an admissible extension and every argument defended by S belongs to S
- S is the *grounded extension* of an AF iff it is the smallest element (with respect to set inclusion) among the complete extensions of the AF.

Intuitively, a GE represents a 'sceptical' solution, consisting only of non-controversial arguments (those not attacking each other).

IV. MARLeME

A diagrammatic summary of how MARLeME can be applied to MARL systems is given in Figure 1, where interpretable models are extracted from MARL system agent data. The three main components of the MARLeME library are described below.

TrajectoryHandler: Agents' trajectories are the agents' data MARLeME operates on. In practice, there is a range of ways in which trajectory data may be provided to MARLeME by the user (referred to as 'raw data' in Figure 1), including different data types (e.g. as text files, JSON objects, or in databases), data loading behaviour (e.g. from a server, or locally), and grouping (e.g. incremental, or batch). The TRAJECTORYHANDLER module is designed to handle these potential variations, before passing the formatted trajectory data to the MODELEXTRACTOR component.

ModelExtractor: The MODELEXTRACTOR component is designed to extract interpretable models from formatted trajectory data provided by the TRAJECTORYHANDLER. This component can utilise extraction algorithms for different types of interpretable models, and provide various underlying implementations for these algorithms (e.g. GPU-optimised). The resulting models are represented using TEAM and AGENT components, which are described below.

Team + Agent: Extracted agent models (shown by green circles in Figure 1) are represented using the TEAMMODEL and AGENTMODEL components. An AGENTMODEL represents a single agent, and a TEAMMODEL represents a group of agents (e.g. teams or sub-teams), giving the user an opportunity to encapsulate agent interactions (e.g. state information sharing). Together, these two classes capture the full spectrum of possible agent interactions: from fully centralised, with all agents represented by a single TEAMMODEL component, to fully decentralised, with all agents running independently (with every agent represented by an AGENTMODEL component). These extracted models can be further analysed (e.g. through manual inspection, formal verification, or statistical analysis) to provide insight into the behaviour of the underlying MARL systems at hand and extract new domain knowledge. Moreover, the extracted models can replace the original uninterpretable ones, providing more interpretable systems with verifiable properties.

MARLeME can be applied to a vast range of existing MARL systems, by simply logging trajectories of the corresponding MARL agents. Furthermore, MARLeME can be used to extract a wide variety of model types, such as decision trees, or fuzzy rule-based models [20], depending on user preferences/needs. This allows successful integration of RL approaches with the well-studied verification, knowledge extraction, and reasoning approaches associated with symbolic models.

A. Abstract Argumentation Agents

The interpretable agents presented in this work (referred to as *AA-based agents*) rely on VAFs for knowledge representation, and GE semantics for action derivation. The arguments used by these agents are *action arguments*, representing action-based heuristics. An action argument A has the form 'If *condition* holds then *agent* should do *action*'. In this definition, *condition* is a boolean function of agent state (specifying whether or not argument A is applicable in that state), *action* specifies the recommended action, and $agent$ refers to an agent in the team. In this work, all AA-based agents have Att defined as follows: argument A attacks argument B iff they recommend the same action to different agents, or different actions to the same agent. For a given agent, arguments in its argument set Arg that recommend actions to that agent are referred to as its *primary arguments* in the rest of this work. In our case, V is the integer set, $Valpref$ is the standard integer ordering, and val is a lookup table, mapping arguments to their integer values.

When deriving an action from a state, an AA-based agent computes all arguments in Arg that are applicable in that state (all arguments for which the argument's condition is true). Then, the agent constructs a VAF from these arguments (using attack construction rules Att, and argument values val). Finally, the agent uses GE semantics to compute the acceptable primary arguments of the VAF. By definition, all arguments acceptable under GE semantics are non-conflicting, thus, given the agent's definition of Att, all acceptable primary arguments must recommend the same action, which is the action executed by the agent.

It should be noted that AA-theory is vast, containing a range of types of AA systems with differing assumptions and flexibility. This work relies on one type of AA-based agent model described above, which is suitable for the corresponding evaluation tasks. Nevertheless, exploring other types of AA-based agents that may give better performance on other types of tasks is an important direction for future work.

B. Abstract Argumentation Agent Model Extraction

Extraction of AA-based agents is performed by the algorithm shown in Algorithm 1. This algorithm assumes the agent argument set (*Args*) is provided by the user, and uses it to derive the argument value ordering (*extractedOrdering*) from the input MARL trajectory data (*Trajectories*), and a user-provided default input ordering (*DefOrder*). The agent models are consequently generated using the user-specified arguments and attack construction rules, together with the derived argument values. The derived value ordering reflects the relative utility of the agent arguments, and can thus serve as an indication of which arguments the agent primarily relies

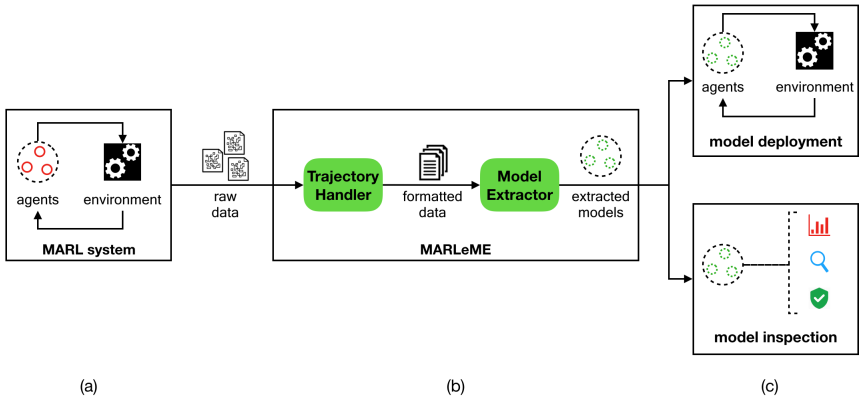

Fig. 1. (a) A MARL system consists of a set of RL agents (red circles) interacting with each other and their environment. (b) MARLeME uses agent trajectory data obtained from the MARL system (raw data), in order to extract a set of interpretable models (green circles) from that data, which approximate the behaviour of the original agents. (c) The interpretable models can replace the original ones (model deployment) or be investigated to better understand the behaviour of the underlying MARL system (model inspection).

Algorithm 1 extractOrdering($Trajectories, Args, DefOrder$)

$prefGraph = newArgPreferenceGraph()$
for $(state, action) \in Trajectories$ **do**
  $applicableArgs = getApplicableArgs(Args, state)$
  $relevantArgs =$
    $\{arg \in applicableArgs \mid$
      $arg.RecAction == action\}$
  $irrelevantArgs = applicableArgs - relevantArgs$
  for $relevantArg \in relevantArgs$ **do**
    for $irrelevantArg \in irrelevantArgs$ **do**
      $prefGraph.incrementEdge(relevantArg,$
        $irrelevantArg)$
    end for
  end for
end for
$DAG = convertToAcyclic(prefGraph)$
$extractedOrdering = topologicalSort(DAG.nodes,$
  $DAG.edges,$
  $DefOrder)$
return $extractedOrdering$

on during action selection, allowing interpretation of agent strategy (as will be shown in Section VI-A2).

Algorithm 1 relies on topological sorting and operates on a novel Argument Preference Graph (APG) structure (also introduced in this work). The fundamental idea behind the algorithm is that for a given pair of arguments (A, B) and a RL agent trajectory, the argument that is 'in closer agreement' with the agent contains relatively more useful information, and should thus have a relatively higher value. These pair-wise argument preferences are stored in an APG. An APG is a weighted directed graph (with non-negative weights) in which the nodes represent arguments, and weighted edges represent preferences between arguments.

The relative argument utility is computed by iterating over all $(state, action)$ pairs of input agent trajectories. For every such $(state, action)$ pair, the APG is updated by incrementing the weights of directed edges between all pairs of arguments (A, B), where both A and B are applicable in $state$, A was in agreement with the agent (recommended the same action), and B was not (recommended a different action). Thus, a directed edge with weight w from argument A to argument B in an APG implies that A was applicable and in agreement with the agent, whilst argument B was applicable and was not in agreement with the agent, for w different trajectory states. A high value of weight w of a directed edge (A, B) thus signifies that argument A frequently suggested more relevant information than argument B, implying that argument A should have a higher value.

Once the APG is constructed, an argument ordering is extracted from it by first converting it into a Directed Acyclic Graph (DAG) by calling the *convertToAcyclic* method, and then topologically sorting the DAG (by calling the *topologicalSort* method). Graph cycle removal is achieved by relying on a pruning heuristic, which removes all edges of weight less than a given pruning value p from the graph. Topological sorting is achieved using a slight variation of Kahn's algorithm [12], where the node extraction order relies on the default argument ordering given by the user (*DefOrder*), instead of relying on stack or queue structures.

Our ordering extraction algorithm relies on both knowledge derived from the trajectory data (in the form of an APG), and on heuristics injected by the user (the default ordering). Graph cycle removal from potentially-cyclic graphs is a highly researched topic, with a wide range of proposed approaches (such as those given in [1], [2]), many of which could potentially be used as implementations of the $convertToAcyclic$ method. The utilised pruning approach is advantageous since it provides a straightforward way of controlling the tradeoff between the importance of prior knowledge and extracted knowledge, by using the pruning parameter p. A higher p value removes more edges and yields a sparser APG with more possible argument orderings, leading to greater reliance

on the default ordering. A lower p value has the reverse effect.

Overall, the extraction algorithm relies on pre-defined attack rules and arguments, focusing on deriving relative argument values from the data. Relying on manually-provided arguments enables expert knowledge to be injected directly into the extracted models, which in turn is very useful when generating the models (i.e. attempting to approximate agents by a set of heuristics experts are familiar with). However, in certain environments it may be desirable to perform argument/attack derivation automatically (e.g. to ensure better scalability). Thus, we intend to explore modifications to the extraction algorithm, allowing automated derivation of arguments/attacks (e.g. through automated rule extraction) in future work.

V. Experiments

We evaluated the AA-based models extracted by MARLeME using two well-known RL case studies: Cooperative Navigation [37] and RoboCup Takeaway [15]. The remainder of this section describes the two case studies, as well as the corresponding AA-based agent setups.

A. Cooperative Navigation

In the Coop. Nav. task, N agents must cooperate through physical actions to reach a set of L landmarks (in our case, $N = L = 2$). Agents observe the relative positions of other agents and landmarks, and are collectively rewarded based on the proximity of any agent to each landmark. Thus, the agents learn to infer the landmarks they must cover, and move there while avoiding other agents. The agents were trained using the multi-agent deep deterministic policy gradient (MADDPG) method (further details can be found in [37]).

We defined the following argument set Arg for the two agents (here i is the landmark index, and j is the agent index, both ranging from 1 to 2):

- $ClosestLandmark_{i,j}$: If L_i is the closest landmark to A_j, A_j should go towards L_i.
- $ClosestAgent_{i,j}$: If A_j is the closest agent to L_i, A_j should go towards L_i.
- $ClosestAgentLandmark_{i,j}$: If A_j is the closest agent to L_i, and L_i is the closest landmark to A_j, A_j should go towards L_i.
- $GoToLandmark_{i,j}$: Agent A_j should go towards L_i

The above argument set is small, easy to work with, and consists of intuitive heurisitcs likely to affect agent behaviour.

B. RoboCup Takeaway

RoboCup Takeaway was used to evaluate MARLeME on a more challenging task, involving a large state space, competing agents, and long, variable delays in action effects. RoboCup Takeaway was proposed in [15] in order to facilitate RL research in the context of RoboCup Soccer [2], and focuses on two teams of simulated agents playing the Takeaway game in a two-dimensional virtual soccer stadium. In Takeaway, $N+1$ hand-coded keepers are competing with N independent learning takers on a fixed-size field. Keepers attempt to keep possession of the ball, whereas takers attempt to win possession of the ball. The game consists of a series of episodes, and an episode ends when the ball goes off the field or any taker gets the ball. A new episode starts immediately with all the players reset. We focus here on the 4v3 Takeaway game, consisting of 4 keepers and 3 takers. The MARL taker team consisted of homogeneous, independent learning RL agents relying on the SARSA(λ) algorithm with tile-coding function approximation [35].

When defining arguments for AA-based taker agents, we made use of the work in [14], which explored RL agent convergence, and relied on Takeaway as the case study. The mentioned works defined a set of arguments containing useful domain heuristics relevant to the takeaway game. Every AA-based taker uses the same argument set Arg, consisting of the following arguments (here i is the taker index, ranging from 1 to 3, and p is the keeper index, ranging from 1 to 4):

- $TackleBall_i$: If T_i is closest to the ball holder, T_i should tackle the ball
- $OpenKeeper_{i,p}$: If a keeper K_p is in a quite 'open' position, T_i should mark this keeper
- $FarKeeper_{i,p}$: If a keeper K_p is 'far' from all takers, T_i should mark this keeper
- $MinAngle_{i,p}$: If the angle between T_i and a keeper K_p, with vertex at the ball holder, is the smallest, T_i should mark this keeper
- $MinDist_{i,p}$: If T_i is closest to a keeper K_p, T_i should mark this keeper

VI. Results

This section presents the results obtained by evaluating the extracted AA-based models on the chosen case studies. Section VI-A and Section VI-B demonstrate how the extracted models can be inspected (both qualitatively and formally), in order to better understand the underlying RL models, and extract useful knowledge from them. Section VI-C demonstrates how well the extracted models perform on their original tasks, and how closely they approximate their original models.

A. Qualitative Inspection

Given that extracted models serve as approximations of original models, they may be used to identify and understand the high-level strategies of the original agents, as well as to explain individual action selection. This can be used to extract knowledge about the domain, the nature of agent interactions, and the types of roles agents can take in a team. Our qualitative inspection of extracted models presented here is two-fold: firstly (Section VI-A1), we inspect the extracted models at the level of action-selection. Secondly (Section VI-A2), we inspect the extracted models at the policy level.

1) Action Selection: The AA-based agents proposed here make use of an interpretable action selection strategy when deriving an action from a state, compared to their original RL models (an example is shown in Figure 2). Any action selected by an AA-based agent in a given state can be easily

[2]https://rcsoccersim.github.io/

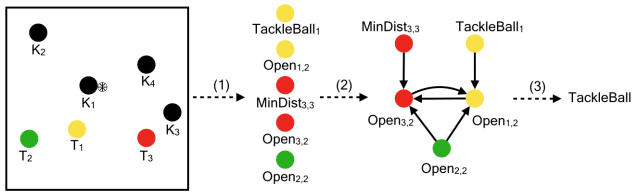

Fig. 2. AA-based Taker Agent Model action selection sub-steps for taker T_1. (1) Determines the applicable agent arguments for all takers T_1, T_2, T_3 (shown by yellow, green, and red circles, respectively), using the input state attributes. (2) Constructs a VAF from the argument set by injecting attacks between arguments (using attack construction rules and the argument value ordering). (3) Derives an action from the VAF for T_1, using GE semantics.

traced back to the original set of defined heuristics (Arg), their interactions (Att), and their relative values (val). As described in Section IV-A, an agent's argument set contains arguments recommending actions to the agent itself, as well as arguments recommending actions to its teammates (as shown in Figure 2). Thus, action selection analysis can be used to study the motivations of an individual agent, as well as inter-agent interactions, that influence action selection.

2) Strategy Analysis: AA-based agent models offer an intuitive way of analysing agent strategies as a whole, exploring individual agent behaviour, and cooperative team behaviour, by using their argument value orderings val. As described in Section IV-A, an agent chooses an action recommended by its primary arguments. Given that arguments with higher values defeat arguments with lower values, high-valued primary arguments represent the main heuristics used by an agent during decision-making. Thus, analysing the highest-valued primary arguments can be used to explore individual agent strategies, and their behaviour. We give an example using the more complex Takeaway task in the remainder of this section.

TABLE I
HIGHEST-VALUED PRIMARY ARGUMENTS OF THE AA-BASED TAKER AGENTS

Agent 1	Agent 2	Agent 3
$TackleBall_1$	$TackleBall_2$	$TackleBall_3$
$MinAngle_{1,3}$	$MinAngle_{2,3}$	$MinAngle_{3,4}$
$MinAngle_{1,2}$	$MinDist_{2,3}$	$MinDist_{3,4}$
$MinDist_{1,2}$	$OpenKeeper_{2,3}$	$OpenKeeper_{3,4}$
$MinDist_{1,3}$	$FarKeeper_{2,3}$	$FarKeeper_{3,4}$

Table I gives the top 5 highest-valued primary arguments for the Taker agents. For all three agents, their *TackleBall* argument has the highest value, implying that tackling the ball when closest to the keeper is of paramount importance for every agent. For Agent 1, the four remaining arguments show that this agent focuses on tackling K_2, or K_3, if it is 'closest' to them (closest by angle, or distance). For Agents 2 and 3, their four remaining arguments all recommend marking K_3 and K_4 (respectively), implying that these agents focus on tackling these keepers throughout the game, preventing the ball-holder from passing to them.

The above analysis demonstrates the specialised roles taken by the different agents (e.g. different agents in the above example focus on tackling different keepers). As discussed previously, this knowledge may be used to study the Takeaway game in more detail, and attempt to synthesise winning agent strategies. For instance, the above example demonstrates that a possible strategy is for one taker to tackle the ball-holder, and the other two to 'spread out' and tackle distant keepers, in order to prevent them from receiving the ball.

B. Formal Inspection

In the previous section we focused on the qualitative aspect of model inspection. A key utility of symbolic models is that their properties can be proven using automated reasoning and formal verification methods. Thus, we now turn our attention to the formal aspect of model inspection, which is of paramount importance in safety and security critical systems.

In Coop. Nav., an important property of agent strategy ensuring good performance is for the agents to target distinct landmarks. We show that our AA-based agents always spread out to different landmarks under minimal assumptions.

Theorem 1. *Assume that every argument in Arg has a unique value. Then, for any state s, agents A_1 and A_2 will always choose distinct landmarks.*

Sketch. By definition, all four $GoToLandmark$ arguments will be applicable in any state s. Thus, there is at least one argument recommending each landmark to either agent. Consider the VAF derived from s. Denote the argument in the VAF with the highest value by Arg_v (value is unique by assumption). Arg_v is not attacked by any other argument, and will therefore be in the derived GE. Denote the landmark recommended by Arg_v as L_v, and the corresponding agent by $Agent_v$. By definition, Arg_v attacks all arguments that recommend the other landmark to $Agent_v$, and that recommend the same landmark to the other agent. Thus, it defends all argument(s) that recommend the other landmark to the other agent (of which there is at least one in the VAF), which will thus also be in the GE. Hence, the GE will contain arguments recommending distinct landmarks to both agents. □

In Takeaway, an important property of taker strategy that increases the chances of the takers intercepting the ball is ensuring that the ball-holder is tackled. This property implies that the ball-holder will be forced to pass the ball to another keeper, giving the takers an opportunity to intercept it. We show that in our team of AA-based takers there is always one taker tackling the ball-holder.

Lemma 1. *Taker T is guaranteed to take action A in state s if the GE derived from this state includes A and A is a primary argument for T.*

Sketch. We know from Section IV-A, that for any taker T and any state s, all acceptable primary arguments in the GE derived from s recommend the same action, which is the action executed by T. □

Lemma 1 guarantees that the heuristics extracted from the expert knowledge are followed faithfully by an agent. Using this result, we can now proceed to proving the desired property.

Theorem 2. *There is always exactly one taker tackling the keeper in possession of the ball in any state, subject to the fact that takers have different values for the TackleBall argument.*

Sketch. Denote (w.l.o.g.) the taker closest to the ball-holder by T. If there are multiple takers at the same distance to the ball-holder, select the one with the largest $TackleBall$ argument value (which exists, by assumption). Denote the state information T receives from the environment by s. By definition, the $TackleBall_T$ argument is applicable in state s, and will thus be included in the VAF derived from s.

By definition of the attack construction rules described in Section IV-A, arguments in the VAF attacking $TackleBall_T$ are: (1) Arguments recommending a different action to T. (2) Arguments recommending the same action to other takers. Firstly, TackleBall is the highest-valued primary argument for every taker (as shown in Table I), including T. Secondly, the only arguments recommending the same action (tackling the ball-holder) to other takers are their respective $TackleBall$ arguments (as discussed in Section V-B), which all have a lower value than $TackleBall_T$ (by assumption).

Hence, $TackleBall_T$ will defeat all arguments attacking it, and will therefore be in the GE derived from the VAF. Thus, by Lemma 1, T will tackle the ball-holder.

□

C. Deployment

Finally, the utility of the extracted models was evaluated quantitatively, by replacing the original RL agents with their corresponding AA-based agents, and comparing the task performance of the two approaches. In case of Coop. Nav., where episodes timeout after a fixed number of timesteps, task performance was measured by recording average reward per episode, over a 1000 episodes (a higher reward signifies better performance). In case of Takeaway, where episodes have a defined termination criterion, task performance was measured by recording average episode duration for 1000 episodes (a shorter episode duration signifies better performance). The above setup was run on a MacBook Pro computer with a 4-core 2.5GHz Intel Core i7 processor, and 16GB of main memory. The performances of the different approaches are shown in Table II.

TABLE II
TASK PERFORMANCE OF THE ORIGINAL AND EXTRACTED MODELS

	Original Models	**Extracted Models**
Coop. Nav. (avg. reward)	-152.42 +/- 2.3	**-141.80 +/- 0.2**
Takeaway (ep. duration)	**9.55 +/- 3.3 s**	10.71 +/- 3.8 s

In case of simpler tasks, interpretable models may encapsulate and use domain-specific assumptions/heuristics directly, achieving higher performance compared to the original models that have to learn these heuristics, as is the case with Coop. Nav. agents. In case of more complex tasks, increased interpretability of extracted models often comes at a cost of their reduced flexibility, implying that they may perform sub-optimal decision making, compared to the original models, and therefore incur a reduction in performance, as is the case with AA-based taker agents.

TABLE III
FIDELITY OF EXTRACTED MODELS

	Fidelity
Coop. Nav.	0.85 (Agent Team)
Takeaway	0.86 (Agent 1), 0.64 (Agent 2), 0.82 (Agent 3)

In addition to evaluating task performance, we also computed the fidelity of extracted models, shown in Table III. Intuitively, closeness of approximation allows to assess the degree to which we can rely on the extracted models for studying/replacing the original ones. For the simpler Coop. Nav. task, we computed fidelity by measuring the percentage of episodes in which both the extracted and original models chose the same landmark allocation for the agents (i.e. fidelity is computed for the agent team). For Takeaway, we used the well-known *0-1 loss* (percentage of actions in which the extracted model agreed with the original one, over a sequence of episodes), which is often used in imitation learning [13] (i.e. fidelity is computed separately for every agent). In case of Coop. Nav., the extracted models achieved a fidelity score of 0.85. In case of the more challenging Takeaway task, agent 1 and agent 3 achieved high fidelity scores, whilst agent 2 achieved a relatively lower fidelity score. These results indicate that model extraction for agent 2 could be further improved by using a more flexible extracted model (e.g. an AA-based agent model with a larger argument set Arg), consequently making extracted model interpretation even more meaningful.

VII. CONCLUSIONS

We introduce MARLeME, a Multi-Agent Reinforcement Learning Model Extraction library, designed to improve interpretability of MARL systems by approximating MARL agents with symbolic models. We also introduce interpretable models based on Abstract Argumentation, discussing how they can be used to approximate MARL systems and used by MARLeME as extracted models.

MARLeME can be applied to a vast range of existing MARL systems, and can be used with a wide variety of symbolic model types. Furthermore, MARLeME can be combined with commonly-used statistical tools (such as scikit-learn [33], or Pandas [34]), when empirically analysing the extracted models, and with formal verification tools (e.g. [10], [11]) when formally inspecting them. Thus, MARLeME allows successful integration of RL approaches with the well-studied verification, knowledge extraction, and reasoning approaches associated with symbolic models. In this work, we focus on using well-known, accessible MARL benchmarks, demonstrating how to apply MARLeME to commonly-used tasks. In order to further highlight the importance of catering for formal

verification methods, as well as statistical ones, we intend to explore applications of MARLeME to safety/security critical domains (e.g. the medical domain) in the future.

With the rapidly increasing interest in MARL systems and the development of associated tools, we believe MARLeME can play a fundamental role in enriching such MARL systems with explainability and interpretability.

REFERENCES

[1] Ana Paulina Figueroa and César Hernández-Cruz and Mika Olsen, "The minimum feedback arc set problem and the acyclic disconnection for graphs", Discrete Mathematics, vol. 340, 2017

[2] Jianer Chen and Fedor V. Fomin and Yang Liu and Songjian Lu and Yngve Villanger, "Improved algorithms for feedback vertex set problems", Journal of Computer and System Sciences, vol. 74, 2008

[3] Riedmiller, Martin and Moore, Andrew and Schneider, Jeff, "Reinforcement Learning for Cooperating and Communicating Reactive Agents in Electrical Power Grids, Balancing Reactivity and Social Deliberation in Multi-Agent Systems", Springer Berlin Heidelberg, pp 137–149, 2001

[4] Tesauro, Gerald and Kephart, Jeffrey O., "Pricing in Agent Economies Using Multi-Agent Q-Learning", Autonomous Agents and Multi-Agent Systems, 2002

[5] Stone, Peter and Veloso, Manuela, "Multiagent Systems: A Survey from a Machine Learning Perspective", Autonomous Robots, 2000

[6] Nguyen, Thanh Thi and Nguyen, Ngoc Duy and Nahavandi, Saeid, "Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications", arXiv preprint arXiv:1812.11794, 2018

[7] Sallab, Ahmad EL and Abdou, Mohammed and Perot, Etienne and Yogamani, Senthil, "Deep reinforcement learning framework for autonomous driving", Electronic Imaging No. 19, Society for Imaging Science and Technology, 2017

[8] Liu, Ying and Logan, Brent and Liu, Ning and Xu, Zhiyuan and Tang, Jian and Wang, Yangzhi, "Deep Reinforcement Learning for Dynamic Treatment Regimes on Medical Registry Data", IEEE International Conference on Healthcare Informatics, 2017

[9] Barto, Andrew G and Thomas, PS and Sutton, Richard S, "Some recent applications of reinforcement learning", Proceedings of the 18th Yale Workshop on Adaptive and Learning Systems, 2017

[10] Pilotto, Concetta, "Systematic Design and Formal Verification of Multi-Agent Systems", California Institute of Technology, 2011

[11] Kouvaros, Panagiotis and Lomuscio, Alessio and Pirovano, Edoardo and Punchihewa, Hashan, "Formal Verification of Open Multi-Agent Systems", Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS, International Foundation for Autonomous Agents and Multiagent Systems, pp 179–187, 2019

[12] Kahn, A. B., "Topological Sorting of Large Networks", Communications ACM, vol. 5, pp 558-562, 1962

[13] Ross, Stephane and Gordon, Geoffrey and Bagnell, Drew, "A reduction of imitation learning and structured prediction to no-regret online learning", Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp 627–635, 2011

[14] Gao, Yang and Toni, Francesca, "Argumentation Accelerated Reinforcement Learning for Cooperative Multi-Agent Systems", ECAI, pp333-338, 2014

[15] Iscen, Atil and Erogul, Umut, "A new perspective to the keepaway soccer: the takers", Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, vol. 3, pp 1341–1344, 2008

[16] Riveret, Regis and Gao, Yang and Governatori, Guido and Rotolo, Antonino and Pitt, Jeremy and Sartor, Giovanni, "A probabilistic argumentation framework for reinforcement learning agents", Autonomous Agents and Multi-Agent Systems, pp 1–59, 2019

[17] Ontanon, Santiago and Plaza, Enric, "An argumentation framework for learning, information exchange, and joint-deliberation in multi-agent systems", Multiagent and Grid Systems, pp 95–108, 2011

[18] Gao, Yang and Toni, Francesca and Wang, Hao and Xu, Fanjiang, "Argumentation-based multi-agent decision making with privacy preserved", Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, pp 1153–1161, 2016

[19] Dung, Phan Minh, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games", Artificial intelligence, vol. 77, pp 321–357, 1995

[20] Benitez, J. M. and Castro, J. L. and Requena, I., "Are Artificial Neural Networks Black Boxes?", 1997

[21] Verma, Abhinav and Murali, Vijayaraghavan and Singh, Rishabh and Kohli, Pushmeet and Chaudhuri, Swarat, "Programmatically interpretable reinforcement learning", arXiv preprint arXiv:1804.02477, 2018

[22] Hussein, Ahmed and Gaber, Mohamed Medhat and Elyan, Eyad and Jayne, Chrisina, "Imitation learning: A survey of learning methods", ACM Computing Surveys (CSUR), vol. 50, 2017

[23] Le, Hoang M and Yue, Yisong and Carr, Peter and Lucey, Patrick, "Coordinated multi-agent imitation learning", Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp 1995–2003, 2017

[24] Hein, Daniel and Udluft, Steffen and Runkler, Thomas A, "Interpretable policies for reinforcement learning by genetic programming", Engineering Applications of Artificial Intelligence, vol. 76, pp 158–169, 2018

[25] Brown, Alexander and Petrik, Marek, "Interpretable Reinforcement Learning with Ensemble Methods", arXiv preprint arXiv:1809.06995, 2018

[26] Garcez, Artur d'Avila and Gori, Marco and Lamb, Luis C and Serafini, Luciano and Spranger, Michael and Tran, Son N, "Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning", arXiv preprint arXiv:1905.06088, 2019

[27] Domingos, Pedro and Lowd, Daniel, "Unifying Logical and Statistical AI with Markov Logic", Communications ACM, vol. 62, pp 74–83, 2019

[28] Li, Yuxi, "Deep reinforcement learning: An overview", arXiv:1701.07274, 2017

[29] Evans, Richard and Grefenstette, Edward, "Learning Explanatory Rules from Noisy Data", J. Artif. Int. Res., vol. 61, pp 1–64, 2018

[30] Noureddine, Dhouha Ben and Gharbi, Atef and Ahmed, Samir Ben, "Multi-agent Deep Reinforcement Learning for Task Allocation in Dynamic Environment", ICSOFT, pp 17–26, 2017

[31] Lin, Kaixiang and Zhao, Renyu and Xu, Zhe and Zhou, Jiayu, "Efficient large-scale fleet management via multi-agent deep reinforcement learning", Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1774–1783, 2018

[32] Huttenrauch, Maximilian and Sosic, Adrian and Neumann, Gerhard, "Guided deep reinforcement learning for swarm systems", arXiv preprint arXiv:1709.06011, 2017

[33] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V.and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P.and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python", Journal of Machine Learning Research, vol. 12, pp 2825–2830, 2011

[34] Wes McKinney, "Data Structures for Statistical Computing in Python", Proceedings of the 9th Python in Science Conference, pp 51–56, 2010

[35] Sutton, Richard S and Barto, Andrew G and others, "Introduction to reinforcement learning", vol. 135, 1998

[36] Osbert Bastani and Carolyn Kim and Hamsa Bastani, "Interpretability via Model Extraction", CoRR, vol., abs/1706.09773, 2017

[37] Ryan Lowe and Yi Wu and Aviv Tamar and Jean Harb and Pieter Abbeel and Igor Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments", 2017

[38] Pablo Hernandez-Leal and Michael Kaisers and Tim Baarslag and Enrique Munoz de Cote, "A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity", 2017

[39] Kaiqing Zhang and Zhuoran Yang and Tamer Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms", 2019

[40] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)", IEEE Access, vol. 6, pp 52138–52160

[41] Anjomshoae, Sule and Najjar, Amro and Calvaresi, Davide and Främling, Kary, "Explainable Agents and Robots: Results from a Systematic Literature Review", Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp 1078–1088